arXiv:1504.03543v1 [cs.LO] 14 Apr 2015

A Definability Dichotomy for Finite Valued CSPs

Anuj Dawar and Pengming Wang

University of Cambridge Computer Laboratory
{anuj.dawar, pengming.wang}@cl.cam.ac.uk

April 15, 2015

Finite valued constraint satisfaction problems are a formalism for describing many natural optimization problems, where constraints on the values that variables can take come with rational weights and the aim is to find an assignment of minimal cost. Thapper and Živný have recently established a complexity dichotomy for finite valued constraint languages. They show that each such language either gives rise to a polynomial-time solvable optimization problem, or to an NP-hard one, and establish a criterion to distinguish the two cases. We refine the dichotomy by showing that all optimization problems in the first class are definable in fixed-point language with counting, while all languages in the second class are not definable, even in infinitary logic with counting. Our definability dichotomy is not conditional on any complexity-theoretic assumption.

1 Introduction

Constraint Satisfaction Problems (CSPs) are a widely-used formalism for describing many problems in optimization, artificial intelligence and many other areas. The classification of CSPs according to their tractability has been a major area of theoretical research ever since Feder and Vardi [8] formulated their dichotomy conjecture. The main aim is to classify various constraint satisfaction problems as either tractable (i.e. decidable in polynomial time) or NP-hard and a number of dichotomies have been established for special cases of the CSP as well as generalizations of it. In particular, Cohen et al. [5] extend the algebraic methods that have been very successful in the classification of CSPs to what they call *soft constraints*, that is constraint problems involving optimization rather than decision problems. In this context, a recent result by Thapper and Živný [12] established a complexity dichotomy for *finite valued CSPs* (VCSPs). This is

a formalism for defining optimization problems that can be expressed as sums of explicitly given rational-valued functions (a more formal definition is given in Section 2). As Thapper and Živný argue, the formalism is general enough to include a wide variety of natural optimization problems. They show that every finite valued CSP is either in P or NP-hard and provide a criterion, in terms of the existence of a definable XOR function, that determines which of the two cases holds.

In this paper we are interested in the *definability* of constraint satisfaction problems in a suitable logic. Definability in logic has been a significant tool for the study of CSPs for many years. A particular logic that has received attention in this context is Datalog, the language of inductive definitions by function-free Horn clauses. A dichotomy of definability has been established in the literature, which shows that every constraint satisfaction problem on a fixed template is either definable in Datalog or it is not definable even in the much stronger C^ω—an infinitary logic with counting. This result has not been published as such but is an immediate consequence of results in [2] where it is shown that every CSP satisfying a certain algebraic condition is not definable in C^ω, and in [3] where it is shown that those that fail to satisfy this condition have bounded width and are therefore definable in Datalog. The definability dichotomy so established does not line up with the (conjectured) complexity dichotomy as it is known that there are tractable CSPs that are not definable in Datalog.

In the context of the definability of optimization problems, one needs to distinguish three kinds of definability. In general an optimization problem asks for a *solution* (which will typically be an assignment of values from some domain D to the variables V of the instance) minimising the value of a *cost* function. This problem is standardly turned into a decision problem by including a budget b in the instance and asking if there is a solution that achieves a cost of at most b. Sentences in a logic naturally define decision problems, and in the context of definability a natural question is whether the decision problem is definable. Asking for a formula that defines an actual optimal solution may not be reasonable as such a solution may not be uniquely determined by the instance and formulas in logic are generally invariant under automorphisms of the structure on which they are interpreted. An intermediate approach is to ask for a term in the logic that defines the *cost* of an optimal solution and this is our approach in this paper.

Our main result is a definability dichotomy for finite valued CSPs. In the context of optimization problems involving numerical values, Datalog is unsuitable so we adopt as our yardstick definability in fixed-point logic with counting (FPC). This is an important logic that defines a natural and powerful proper fragment of the polynomial-time decidable properties (see [6]). It should be noted that C^ω properly extends the expressive power of FPC and therefore undefinability results for the former yield undefinability results for the latter. We establish that every finite valued CSP is either definable in FPC or undefinable in C^ω. Moreover, this dichotomy lines up exactly with the complexity dichotomy of Thapper and Živný. All the valued CSPs they determine are tractable are in fact definable in FPC, and all the ones that are NP-hard are provably not in C^ω. It should be emphasised that, unlike the complexity dichotomy, our definability dichotomy is not conditional on any complexity-theoretic assumption. Even if it were the case that P = NP, the finite valued CSPs still divide into those definable in FPC and those that

are not on these same lines.

The positive direction of our result builds on the recent work of Anderson et al. [1] showing that solutions to explicitly given instances of linear programming are definable in FPC. Thapper and Živný show that for the tractable VCSPs the optimal solution can be found by solving their basic linear programming (BLP) relaxation. Thus, to establish the definability of these problems in FPC it suffices to show that the reduction to the BLP is itself definable in FPC, which we do in Section 4.

For the negative direction, we use the reductions used in [12] to establish NP-hardness of VCSPs and show that these reductions can be carried out within FPC. We start with the standard CSP form of 3-SAT, which is not definable in C^ω as a consequence of results from [2]. Details of all these reductions are presented in Section 5.

There is one issue with regard to the representation of instances of VCSPs as relational structures which we need to consider in the context of definability. An instance is defined over a language which consists of a set Γ of functions from a finite domain D to the rationals. If Γ is a finite set, it is reasonable to fix the relational signature to have a relation for each function in Γ, and the FPC formula defining the class of VCSPs would be in this fixed relational signature. Indeed, the result of Thapper and Živný [12] is stated for infinite sets Γ but is really about finite subsets of it. That is, they show that if Γ does not have the XOR property, then *every* finite subset of Γ determines a tractable VCSP and that if Γ does have the XOR property then it contains a finite subset Γ' such that VCSP(Γ') is NP-hard. Our definability dichotomy replicates this precisely. However, we can also consider the *uniform* definability of VCSP(Γ) when Γ is infinite (note that only finitely many functions from the language Γ are used in constraints in any instance). A natural way to represent this is to allow the functions themselves to be elements of the relational structure coding an instance. We can show that our dichotomy holds even under this uniform representation. For simplicity of exposition, we present the results for finite Γ and then, in Section 6, we explain how the proof can be modified to the uniform case where the functions are explicitly given as elements of the structure.

2 Background

Notation. We write $\mathbb{N}$ for the natural numbers, $\mathbb{Z}$ for the integers, $\mathbb{Q}$ for the rational numbers and $\mathbb{Q}^+$ to denote the positive rationals.

We use bars $\bar{v}$ to denote vectors. A vector over a set A indexed by a set I is a function $\bar{v} : I \to A$. We write v_a for $\bar{v}(a)$. Often, but not always, the index set I is $\{1, \ldots, d\}$, an initial segment of the natural numbers. In this case, we also write $|\bar{v}|$ for the length of $\bar{v}$, i.e. d. A matrix M over A indexed by two sets I, J is a function $M : I \times J \to A$. We use the symbol $\dot\cup$ for the disjoint union operator on sets.

If $\bar{v}$ is an I-indexed vector over A and $f : A \to B$ is a function, we write $f(\bar{v})$ to denote the I-indexed vector over B obtained by applying f componentwise to $\bar{v}$.

2.1 Valued Constraint Satisfaction

We begin with the basic definitions of valued constraint satisfaction problems. These definitions are based, with minor modifications, on the definitions given in [12].

Definition 1. *Let D be a finite domain. A* valued constraint language Γ *over D is a set of functions, where each $f \in \Gamma$ has an associated arity $m = \mathrm{ar}(f)$ and $f : D^m \to \mathbb{Q}^+$.*

Definition 2. *An instance of the* valued constraint satisfaction problem (VCSP) *over a valued constraint language Γ is a pair $I = (V, C)$, where V is a finite set of* variables *and C is a finite set of* constraints*. Each constraint in C is a triple (σ, f, q), where $f \in \Gamma$, $\sigma \in V^{\mathrm{ar}(f)}$ and $q \in \mathbb{Q}$.*

A solution *to an instance I of* VCSP(Γ) *is an assignment $h : V \to D$ of values in D to the variables in V. The* cost *of the solution h is given by $cost_I(h) := \sum_{(\sigma,f,q)\in C} q \cdot f(h(\sigma))$. The valued constraint satisfaction problem is then to find a solution with minimal cost.*

In the decision version *of the problem, an additional threshold constant $t \in \mathbb{Q}$ is given, and the question becomes whether there is a solution h with $cost_I(h) \leq t$.*

Given a valued constraint language Γ, there are certain natural closures Γ' of this set of functions for which the computational complexity of VCSP(Γ) and VCSP(Γ') coincide. The first we consider is called the *expressive power* of Γ, which consists of functions that can be defined by minimising a cost function over a fixed VCSP(Γ)-instance I over some projection of the variables in I (this is defined formally below). The second closure of Γ we consider is under scaling and translation. Both of these are given formally in the following definition.

Definition 3. *Let Γ be a valued constraint language over D. We say a function $f : D^m \to \mathbb{Q}$, is* expressible *in Γ, if there is some instance $I_f = (V_f, C_f) \in$ VCSP(Γ) and a tuple $\bar{v} = (v_1, \ldots, v_m) \in V_f^m$ such that*

$$f(\bar{x}) = \min_{h \in H_{\bar{x}}} cost_{I_f}(h),$$

where $H_{\bar{x}} := \{h : V_f \to D \mid h(v_i) = x_i \ , \ 1 \leq i \leq m\}$. We then say the function f is expressed *by the instance I_f and the tuple $\bar{v}$, and call the set of all functions that can be expressed by an instance of* VCSP(Γ) *the* expressive power *of Γ, denoted by $\langle\Gamma\rangle$.*

Furthermore, we write $f' \equiv f$ if f' is obtained from f by scaling *and* translation*, i.e. there are $a, b \in \mathbb{Q}, a > 0$ such that $f' = a \cdot f + b$. For a valued constraint language Γ, we write $\Gamma_{\equiv}$ to denote the set $\{f' \mid f' \equiv f$ for some $f \in \Gamma\}$.*

The next two lemmas establish that closing Γ under these operations does not change the complexity of the corresponding problem. The first of these is implicit in the literature, and we prove a stronger version of it in Lemma 13.

Lemma 4. *Let Γ and Γ' be valued constraint languages on domain D such that $\Gamma' \subseteq \Gamma_{\equiv}$. Then* VCSP($\Gamma'$) *is polynomial-time reducible to* VCSP(Γ)*.*

Lemma 5 (Theorem 3.4, [5]). *Let* Γ *and* Γ' *be valued constraint languages on domain* D *such that* $\Gamma' \subseteq \langle\Gamma\rangle$. *Then* VCSP($\Gamma'$) *is polynomial-time reducible to* VCSP(Γ).

In the study of constraint satisfaction problems, and of structure homomorphisms more generally the *core* of a structure plays an important role. The corresponding notion for valued constraint languages is given in the following definition.

Definition 6. *We call a valued constraint language* Γ *over domain* D *a* core *if for for all* $a \in D$, *there is some instance* $I_a \in \text{VCSP}(\Gamma)$ *such that in every minimal cost solution over* I_a, *some variable is assigned* a. *A valued constraint language* Γ' *over a domain* $D' \subseteq D$ *is a* sub-language *of* Γ *if it contains exactly the functions of* Γ *restricted to* D'. *We say* Γ' *is a* core of Γ, *if* Γ' *is a sub-language of* Γ *and also a core.*

Lemma 7 (Lemma 2.4, [12]). *Let* Γ' *be a core of* Γ. *Then,* $\min_h cost_I(h) = \min_h cost_{I'}(h)$ *for all* $I \in \text{VCSP}(\Gamma)$ *and* $I' \in \text{VCSP}(\Gamma')$ *where* I' *is obtained from* I *by replacing each function of* Γ *by its restriction in* Γ'.

Finally, we consider the closure of Γ under parameterized definitions. That is, we define Γ_c, the language obtained from Γ by allowing functions that are obtained from those in Γ by fixing some parameters.

Definition 8. *Let* Γ *be a core over* D, *we denote by* Γ_c *the language that contains exactly those functions* $f : D^m \to \mathbb{Q}$ *for which there exists*

- *a function* $g \in \Gamma$, *with* $g : D^n \to \mathbb{Q}$ *with* $n \geq m$,
- *an injective mapping* $s_f : \{1, \ldots, m\} \to \{1, \ldots, n\}$,
- *an index set* $T_f \subseteq \{1, \ldots, n\}$,
- *and a partial assignment* $t_f : T_f \to D$,

such that f *is* g *restricted on* t_f, *i.e.* $f(x_{s_f(1)}, \ldots, x_{s_f(m)}) = f(t(x_1), \ldots, t(x_n))$, *where* $t(x_i) = t_f(i)$ *if* $i \in T_f$, *and* $t(x_i) = x_i$ *otherwise. Furthermore, we fix a mapping* $\gamma : \Gamma_c \to \Gamma$ *that assigns each* $f \in \Gamma_c$ *a function* $g = \gamma(f) \in \Gamma$ *with the above properties.*

For example, if $f(x_1, x_2, x_3) \in \Gamma$, then $g(x_1, x_2) := f(x_1, a, x_2)$ for $a \in D$ is in Γ_c.

2.2 Linear Programming

Definition 9. *Let* $\mathbb{Q}^V$ *be the rational Euclidean space indexed by a set* V. *A* linear optimization problem *is given by a* constraint matrix $A \in \mathbb{Q}^{C \times V}$ *and vectors* $\bar{b} \in \mathbb{Q}^C, \bar{c} \in \mathbb{Q}^V$. *Let* $P_{A,\bar{b}} := \{\bar{x} \in \mathbb{Q}^V | A\bar{x} \leq \bar{b}\}$ *be the set of* feasible solutions. *The linear optimization problem is then to determine either that* $P_{A,\bar{b}} = \emptyset$, *or to find a vector* $\bar{y} = \text{argmax}_{\bar{x} \in P_{A,\bar{b}}} \bar{c}^T \bar{x}$, *or to determine that* $\max_{\bar{x} \in P_{A,\bar{b}}} \bar{c}^T \bar{x}$ *is unbounded.*

We speak of the integer linear optimization problem, *if the set of feasible solutions is instead defined as* $P_{A,\bar{b}} := \{\bar{x} \in \mathbb{Z}^V | A\bar{x} \leq \bar{b}\}$.

In the decision version of the problem, an additional constant $t \in \mathbb{Q}$ *is given, and the task is determine whether there exists a feasible solution* $\bar{x} \in P_{A,\bar{b}}$, *such that* $\bar{c}^T \bar{x} \geq t$.

It is often convenient to describe the linear optimization problem $(A, \bar{b}, \bar{c})$ as a system of linear inequalities $A\bar{x} \leq \bar{b}$ along with the objective $\max_{\bar{x} \in P_{A,\bar{b}}} \bar{c}^T \bar{x}$. We may also alternatively, describe an instance with a minimization objective. It is easy to see that such a system can be converted to the standard form of Defintion 9.

Let Γ now be a valued constraint language over D, and let $I = (V, C)$ be an instance of VCSP(Γ). We associate with I the following linear optimization problem in variables $\lambda_{c,\nu}$ for each $c \in C$ with $c = (\sigma, f, q)$ and $\nu \in D^{\mathrm{ar}(f)}$, and $\mu_{x,a}$ for each $x \in V$ and $a \in D$.

$$\min \sum_{c \in C} \sum_{\nu \in D^{\mathrm{ar}(f)}} \lambda_{c,\nu} \cdot q \cdot f(\nu) \quad \text{where } c = (\sigma, f, q) \tag{1}$$

subject to the following constraints.
For each $c \in C$ with $c = (\sigma, f, q)$, each i with $1 \leq i \leq \mathrm{ar}(f)$ and each $a \in D$, we have

$$\sum_{\nu \in D^{\mathrm{ar}(f)} : \nu_i = a} \lambda_{c,\nu} \;=\; \mu_{\sigma_i, a}; \tag{2}$$

for each $x \in V$, we have

$$\sum_{a \in D} \mu_{x,a} \;=\; 1; \tag{3}$$

and for all variables $\lambda_{c,\nu}$ and $\mu_{x,a}$ we have

$$0 \leq \lambda_{c,\nu} \leq 1 \quad \text{and} \quad 0 \leq \mu_{x,a} \leq 1. \tag{4}$$

A feasible *integer* solution to the above system defines a solution $h : V \to D$ to the instance I, given by $h(x) = a$ iff $\mu_{x,a} = 1$. Equations 2 then ensure that $\lambda_{c,\nu} = 1$ for $c = (\sigma, f, q)$ just in case $h(\sigma) = \nu$. Thus, it is clear that an optimal integer solution gives us an optimal solution to I.

If we consider *rational* solutions instead of integer ones, we obtain the *basic LP-relaxation* of I, which we denote BLP(I). The following theorem characterises for which languages Γ BLP(I) has the same optimal solutions as I.

For the statement of the dichotomy result from [12], we need to introduce an additional notion. We say the property (XOR) holds for a valued constraint language Γ over domain D if there are $a, b \in D, a \neq b$, such that $\langle\Gamma\rangle$ contains a binary function f with argmin $f = \{(a, b), (b, a)\}$.

Theorem 10 (Theorem 3.3, [12])**.** *Let Γ be a core over some finite domain D.*

- *Either for each instance I of* VCSP(Γ)*, the optimal solutions of I are the same as* BLP(I)*;*
- *or property* (XOR) *holds for Γ_c and* VCSP(Γ) *is* NP*-hard.*

2.3 Logic

A relational *vocabulary* (also called a *signature* or a *language*) τ is a finite sequence of relation and constant symbols $(R_1, \ldots, R_k, c_1, \ldots, c_l)$, where every relation symbol R_i has a fixed *arity* $a_i \in \mathbb{N}$. A structure $\mathbf{A} = (\mathrm{dom}(\mathbf{A}), R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}}, c_1^{\mathbf{A}}, \ldots, c_l^{\mathbf{A}})$ over the signature τ (or a τ-*structure*) consists of a non-empty set $\mathrm{dom}(\mathbf{A})$, called the *universe* of $\mathbf{A}$, together with relations $R_i^{\mathbf{A}} \subseteq \mathrm{dom}(\mathbf{A})^{a_i}$ and constants $c_j^{\mathbf{A}} \in \mathrm{dom}(\mathbf{A})$ for each $1 \leq i \leq k$ and $1 \leq j \leq l$. Members of the set $\mathrm{dom}(\mathbf{A})$ are called the *elements* of $\mathbf{A}$ and we define the *size* of $\mathbf{A}$ to be the cardinality of its universe.

2.3.1 Fixed-point Logic with Counting

Fixed-point logic with counting (FPC) is an extension of inflationary fixed-point logic with the ability to express the cardinality of definable sets. The logic has two sorts of first-order variable: *element variables*, which range over elements of the structure on which a formula is interpreted in the usual way, and *number variables*, which range over some initial segment of the natural numbers. We write element variables with lower-case Latin letters $x, y, \ldots$ and use lower-case Greek letters $\mu, \eta, \ldots$ to denote number variables.

The atomic formulas of FPC$[\tau]$ are all formulas of the form $\mu = \eta$ or $\mu \leq \eta$, where μ, η are number variables; $s = t$ where s, t are element variables or constant symbols from τ; and $R(t_1, \ldots, t_m)$, where each t_i is either an element variable or a constant symbol and R is a relation symbol (i.e. either a symbol from τ or a relational variable) of arity m. Each relational variable of arity m has an associated type from $\{\mathrm{elem}, \mathrm{num}\}^m$. The set FPC$[\tau]$ of FPC *formulas* over τ is built up from the atomic formulas by applying an inflationary fixed-point operator $[\mathbf{ifp}_{R,\bar{x}}\phi](\bar{t})$; forming *counting terms* $\#_x\phi$, where ϕ is a formula and x an element variable; forming formulas of the kind $s = t$ and $s \leq t$ where s, t are number variables or counting terms; as well as the standard first-order operations of negation, conjunction, disjunction, universal and existential quantification. Collectively, we refer to element variables and constant symbols as *element terms*, and to number variables and counting terms as *number terms*.

For the semantics, number terms take values in $\{0, \ldots, n\}$, where $n = \mathrm{dom}(\mathbf{A})$ and element terms take values in $\mathrm{dom}(\mathbf{A})$. The semantics of atomic formulas, fixed-points and first-order operations are defined as usual (c.f., e.g., [7] for details), with comparison of number terms $\mu \leq \eta$ interpreted by comparing the corresponding integers in $\{0, \ldots, n\}$. Finally, consider a counting term of the form $\#_x\phi$, where ϕ is a formula and x an element variable. Here the intended semantics is that $\#_x\phi$ denotes the number (i.e. the element of $\{0, \ldots, n\}$) of elements that satisfy the formula ϕ. For a more detailed definition of FPC, we refer the reader to [7, 10].

We also consider C^ω—the infinitary logic with counting, and finitely many variables. We will not define it formally (the interested reader may consult [11]) but we need the following two facts about it: its expressive power properly subsumes that of FPC, and it is closed under FPC-reductions, defined below.

It is known by the Immerman-Vardi theorem [7] that fixed-point logic can express

all polynomial-time properties of finite ordered structures. It follows that in FPC we can express all polynomial-time relations on the number domain. In particular, we have formulas with free number variables α, β for defining sum and product, and we simply write $\alpha + \beta$ and $\alpha \cdot \beta$ to denote these formulas. For a number term α and a non-negative integer m, we write $\alpha = m$ as short-hand for the formula that says that α is exactly m. We write $\mathrm{BIT}(\alpha, \beta)$ to denote the formula that is true just in case the β-th bit in the binary expansion of α is 1. Finally, for each constant c, we assume a formula $\mathrm{MULT}_c(W, x, y)$ which works as follows. If B is an ordered set and $W \subseteq B$ is a unary relation that codes the binary representation of an integer w, then MULT_c defines a binary relation $R \subseteq B^2$ which on the lexicographic order on B^2 defines the binary representation of $c \cdot w$.

2.3.2 Reductions

We frequently consider ways of defining one structure within another in some logic L, such as first-order logic or FPC. Consider two signatures σ and τ and a logic L. An *m-ary* L*-interpretation of τ in σ* is a sequence of formulae of L in vocabulary σ consisting of: (i) a formula $\delta(\bar{x})$; (ii) a formula $\varepsilon(\bar{x}, \bar{y})$; (iii) for each relation symbol $R \in \tau$ of arity k, a formula $\phi_R(\bar{x}_1, \ldots, \bar{x}_k)$; and (iv) for each constant symbol $c \in \tau$, a formula $\gamma_c(\bar{x})$, where each $\bar{x}$, $\bar{y}$ or $\bar{x}_i$ is an m-tuple of free variables. We call m the *width* of the interpretation. We say that an interpretation Θ associates a τ-structure $\mathbf{B}$ to a σ-structure $\mathbf{A}$ if there is a surjective map h from the m-tuples $\{\bar{a} \in \mathrm{dom}(\mathbf{A})^m \mid \mathbf{A} \models \delta[\bar{a}]\}$ to $\mathbf{B}$ such that:

- $h(\bar{a}_1) = h(\bar{a}_2)$ if, and only if, $\mathbf{A} \models \varepsilon[\bar{a}_1, \bar{a}_2]$;
- $R^{\mathbf{B}}(h(\bar{a}_1), \ldots, h(\bar{a}_k))$ if, and only if, $\mathbf{A} \models \phi_R[\bar{a}_1, \ldots, \bar{a}_k]$;
- $h(\bar{a}) = c^{\mathbf{B}}$ if, and only if, $\mathbf{A} \models \gamma_c[\bar{a}]$.

Note that an interpretation Θ associates a τ-structure with $\mathbf{A}$ only if ε defines an equivalence relation on $\mathrm{dom}(\mathbf{A})^m$ that is a congruence with respect to the relations defined by the formulae ϕ_R and γ_c. In such cases, however, $\mathbf{B}$ is uniquely defined up to isomorphism and we write $\Theta(\mathbf{A}) := \mathbf{B}$. Throughout this paper, we will often use interpretations where ε is simply defined as the usual equality on $\bar{a}_1$ and $\bar{a}_2$. In these instances, we omit the explicit definition of ε.

The notion of interpretations is used to define logical reductions. Let C_1 and C_2 be two classes of σ- and τ-structures respectively. We say C_1 L*-reduces* to C_2 if there is an L-interpretation Θ of τ in σ, such that $\Theta(\mathbf{A}) \in C_2$ if and only if $\mathbf{A} \in C_1$, and we write $C_1 \leq_{\mathrm{L}} C_2$.

It is not difficult to show that formulas of FPC compose with reductions in the sense that, given an interpretation Θ of τ in σ and a σ-formula ϕ, we can define a τ-formula ϕ' such that $\mathbf{A} \models \phi'$ if, and only if, $\Theta(\mathbf{A}) \models \phi$. Moreover C^ω is closed under FPC-reductions. So if C_2 is definable in C^ω and $C_1 \leq_{\mathrm{L}} C_2$, then C_1 is also definable in C^ω.

2.3.3 Representation

In order to discuss definability of constraint satisfaction and linear programming problems, we need to fix a representation of instances of these problems as relational structures. Here, we describe the representation we use.

Numbers and Vectors. We represent an integer z as a relational structure in the following way. Let $z = s \cdot x$, with $s \in \{-1, 1\}$ being the sign of z, and $x \in \mathbb{N}$, and let $b \geq \lceil \log_2(x) \rceil$. We represent z as the structure $\mathbf{z}$ with universe $\{1, \ldots, b\}$ over the vocabulary $\tau_{\mathbb{Z}} = \{X, S, <\}$, where $<$ is interpreted the usual linear order on $\{1, \ldots, b\}$; $S^{\mathbf{z}}$ is a unary relation where $S^{\mathbf{z}} = \emptyset$ indicates that $s = 1$, and $s = -1$ otherwise; and $X^{\mathbf{z}}$ is a unary relation that encodes the bit representation of x, i.e. $X^{\mathbf{z}} = \{k \in \{1, \ldots, b\} \mid \mathrm{BIT}(x, k) = 1\}$. In a similar vein, we represent a rational number $q = s \cdot \frac{x}{d}$ by a structure $\mathbf{q}$ over the domain $\tau_{\mathbb{Q}} = \{X, D, S, <\}$, where the additional relation $D^{\mathbf{q}}$ encodes the binary representation of the denominator d in the same way as before.

In order to represent vectors and matrices over integers or rationals, we have multi-sorted universes. Let T be a non-empty set, and let v be a vector of integers indexed by T. We represent v as a structure $\mathbf{v}$ with a two-sorted universe with an index sort T, and bit sorts $\{1, \ldots, b\}$, where $b \geq \lceil \log_2(|m|) \rceil$, $m = \max_{t \in T} v_t$, over the vocabulary $(X, D, S, <)$. Now, the relation S is of arity 2, and $S^{\mathbf{v}}(t, \cdot)$ encodes the sign of the integer v_t for $t \in T$. Similarly, X is a binary relation interpreted as $X^{\mathbf{v}} = \{(t, k) \in T \times \{1, \ldots, b\} \mid \mathrm{BIT}(v_t, k) = 1\}$. In order to represent matrices $M \in \mathbb{Z}^{T_1 \times T_2}$, indexed by two sets T_1, T_2, we allow three-sorted universes with two sorts of index sets. The generalisation to rationals carries over from the numbers case. We write τ_{vec} to denote the vocabulary for vectors over $\mathbb{Q}$ and τ_{mat} for the vocabulary for matrices over $\mathbb{Q}$.

Linear Programs. Let an instance of a linear optimization problem be given by a constraint maxtrix $A \in \mathbb{Q}^{C \times V}$, and vectors $\bar{b} \in \mathbb{Q}^C, \bar{c} \in \mathbb{Q}^V$ over some set of variables V and constraints C. We represent this instance in the natural way as a structure over the vocabulary $\tau_{\mathrm{LP}} = \tau_{vec} \dot{\cup} \tau_{mat}$.

We can now state the result from [1] that we require, to the effect that there is an FPC interpretation that can define solutions to linear programs.

Theorem 11 (Theorem 11, [1])**.** *Let an instance $(A \in \mathbb{Q}^{C \times Q}, \bar{b} \in \mathbb{Q}^C, \bar{c} \in \mathbb{Q}^V)$ of a LP be explicitly given by a relational representation in τ_{LP}. Then, there is a* FPC*-interpretation that defines a representation of $(f \in \mathbb{Q}, \bar{v} \in \mathbb{Q}^V)$, such that $f = 1$ if and only if $\max_{\bar{x} \in P_{A,\bar{b}}} \bar{c}^T \bar{x}$ is unbounded, $\bar{v} \notin P_{A,\bar{b}}$ if and only if there is no feasible solution, and $f = 0, \bar{v} = \mathrm{argmax}_{\bar{x} \in P_{A,\bar{b}}} \bar{c}^T \bar{x}$ otherwise.*

CSPs. We next examine how instances of VCSP(Γ) for finite Γ are represented as relational structures. We return to the case of infinite Γ in Section 6.

For a fixed finite language $\Gamma = \{f_1, \ldots, f_k\}$, we represent an instance I of VCSP(Γ) as a structure $\mathbf{I} = (\mathrm{dom}(\mathbf{I}), <, (R_f^{\mathbf{I}})_{f \in \Gamma}, W_N^{\mathbf{I}}, W_D^{\mathbf{I}})$ over the vocabulary τ_Γ. The universe $\mathrm{dom}(\mathbf{I}) = V \dot{\cup} C \dot{\cup} B$ is a three-sorted set, consisting of variables V, constraints C, and a set B of *bit positions*. We assume that $|B|$ is at least as large as the number of bits required to represent the numerator and denominator of any rational weight occurring in I. The relation $<$ is a linear order on B. The relation $R_f^{\mathbf{I}} \subseteq V^{ar(f)} \times C$ contains

(σ, c) if $c = (\sigma, f, q)$ is a constraint in I. The relations $W_N^{\mathbf{I}}, W_D^{\mathbf{I}} \subseteq C \times B$ encode the weights of the constraints: $W_N^{\mathbf{I}}(c, \beta)$ (or $W_D^{\mathbf{I}}(c, \beta)$) holds if and only if the β-th bit of the bit-representation of the numerator (or denominator, respectively) of the weight of constraint c is one. For the decision version of the VCSP, we have two additional unary relation T_N and T_D in the vocabulary which encode the binary representation of the numerator and denominator of the threshold constant of the instance.

We are now ready to define what it means to express VCSP(Γ) in a logic such as FPC. For a fixed finite langauge Γ, we say that the decision version of VCSP(Γ) is definable in a logic L if there is some $\tau_\Gamma \cup \{T_N, T_D\}$-sentence ϕ of L such that $\mathbf{I} \models \phi$ if, and only if, I is satisfiable. We say that VCSP(Γ) is definable in FPC if there is an FPC interpretaion Θ of the vocabulary $\tau_\mathbb{Q}$ in τ_Γ such that for any $\mathbf{I}$, $\Theta(\mathbf{I})$ codes the value of an optimal solution for the instance I.

3 Definable Reductions

An essential part of the machinery that leads to Theorem 10 is that the computational complexity of VCSP(Γ) is robust under certain changes to Γ. In other words, closing the class of functions Γ under certain natural operations does not change the complexity of the problem. This is established by showing that the distinct problems obtained are inter-reducible under polynomial-time reductions. Our aim in this section is to show that these reductions can be expressed as interpretations in a suitable logic (in some cases first-order logic suffices, and in others we need the power of counting).

The following lemma is analogous to Lemma 5 and shows that the reductions there can be expressed as logical interpretations.

Lemma 12. *Let Γ and Γ' be valued constraint languages over domain D of finite sizes such that $\Gamma' \subseteq \langle\Gamma\rangle$. Then* VCSP($\Gamma'$) $\leq_{\text{FPC}}$ VCSP(Γ).

Proof. The construction of the reduction follows closely the proof of Theorem 3.4. in [5], while ensuring it is definable in FPC.

Let $I = (V, C)$ be a given instance of VCSP(Γ'). We fix for each function $f \in \Gamma'$ of arity m an instance $I_f = (V_f, C_f)$ of VCSP(Γ) and a m-tuple of distinct elements $\bar{v}_f \in V_f^m$ that together express f in the sense of Definition 3. The idea is now to replace each constraint $c = (\sigma, f, q) \in C$ by a copy of I_f where the variables $v_{f1}, \ldots, v_{fm}$ in I_f are identified with $\sigma_1, \ldots, \sigma_m$, and the remaining variables are fresh. Since each I_f is an instance of VCSP(Γ), the instance $J = (U, E)$ obtained after all replacements is again an instance of VCSP(Γ). Furthermore, by Definition 3 it has the same optimal solution as I.

Formally, we define the instance $J = (U, E)$ as follows. The set of variables U consists of the variables in V plus a fresh copy of the variables in V_f for each constraint in C that uses the function f, so we can identify U with the following set.

$$U = V \mathbin{\dot\cup} \{(v, c) \mid c \in C, v \in V_f\}.$$

Each constraint $c = (\sigma, f, q) \in C$ gives rise to a set of constraints E_c, representing a copy of the constraints in C_f.

$$E_c = \{(h_c(\nu), g, q \cdot r) \mid (\nu, g, r) \in C_f\},$$

where $h_c : V_f \to U$ is defined as the mapping $h_c(v) = \sigma_i$, if $v = v_{fi}$, and $h_c(v) = (v, c)$ otherwise. The set of constraints E is then simply the union of all sets E_c.

$$E = \bigcup_{c \in C} E_c.$$

Let $\tau_\Gamma = (<, (R_f)_{f\in\Gamma}, W_N, W_D)$ and $\tau_{\Gamma'} = (<, (R_f)_{f\in\Gamma'}, W_N, W_D)$ be the vocabularies for instances of VCSP(Γ) and VCSP(Γ') respectively. We aim to define an FPC reduction $\Theta = (\bar{\delta}, \varepsilon, \phi_<, (\phi_{R_f})_{f\in\Gamma}, \phi_{W_N}, \phi_{W_D})$ such that $\mathbf{J} = \Theta(\mathbf{I})$ corresponds to the above construction of the instance J.

Let an instance $I = (V, C)$ of VCSP(Γ') be given as a structure $\mathbf{I}$ over $\tau_{\Gamma'}$ with the three-sorted universe $\mathrm{dom}(\mathbf{I}) = V \dot\cup C \dot\cup B$. For each m-ary function $f \in \Gamma$ we have fixed an instance $I_f = (V_f, C_f)$ and a tuple $\bar{v}_f = (v_{f1}, \ldots, v_{fm})$ that together express f. As the construction of $\mathbf{J}$ depends on these instances, we fix an encoding of them in an initial segment of the natural numbers. To be precise, as the sets $\hat{V} = \bigcup_{f\in\Gamma'} V_f$ and $\hat{C} = \bigcup_{f\in\Gamma'} C_f$ are of fixed size (independent of I), let $n_{\hat{V}} = |\hat{V}|$ and $n_{\hat{C}} = |\hat{C}|$. We then fix bijections $var : \hat{V} \to \{1, \ldots, n_{\hat{V}}\}$ and $con : \hat{C} \to \{1, \ldots, n_{\hat{C}}\}$ such that for each $f \in \Gamma'$, there are intervals $\mathcal{V}_f = [lv_f, rv_f]$ and $\mathcal{C}_f = [lc_f, rc_f]$ such that $var(V_f) = \mathcal{V}_f$ and $con(C_f) = \mathcal{C}_f$. We assume that $\mathrm{dom}(\mathbf{I})$ is larger than $\max(n_{\hat{V}}, n_{\hat{C}})$ so that we can use number terms to index the elements of $\hat{V}$ and $\hat{C}$. There are only finitely many instances I smaller than this, and they can be handled in the interpretation Θ individually.

In defining the formulas below, for an integer interval I we write $\mu \in I$ as shorthand for the formula $\bigvee_{m\in I} \mu = m$.

The universe of $\mathbf{J}$ is a three-sorted set $\mathrm{dom}(\mathbf{J}) = U \dot\cup E \dot\cup B'$ consisting of variables U, constraints E, and bit positions B'. The set U is defined by the formula

$$\delta_U(x, \mu) = \left(x \in C \wedge \bigvee_{f\in\Gamma'} (\exists \bar{y} \in V^{ar(f)} : R_f(\bar{y}, x) \wedge \mu \in \mathcal{V}_f) \right) \vee (\mu = 0 \wedge x \in V).$$

In other words, the elements of U consist of pairs (x, μ), where $x \in V \cup C$ and μ is an element of the number domain and we make the following case distinction: Either $x \in C$ and there is a constraint $x = (\bar{y}, f, q)$ in I, and a variable $v \in V_f$ with $var(v) = \mu$; then the pair represents one of the fresh variables in $C \times \hat{V}$. Or, $x \in V$ and $\mu = 0$ and the pair simply represents an element of V.

Similarly, the constraints E are given by

$$\delta_E(x, \mu) = x \in C \wedge \bigvee_{f\in\Gamma'} (\exists \bar{y} \in V^{ar(f)} : R_f(\bar{y}, x) \wedge \mu \in \mathcal{C}_f).$$

Again, the elements of E are pairs (x, μ), with $\in C$ and μ an element of the number domain, and we require that if there is a constraint of the form $x = (\bar{y}, f, q)$, then there is a constraint $c \in C_f$ with $con(c) = \mu$.

For the domain of bit positions, we just need to make sure that the set is large enough to encode all weights in J. Taking $B' = B^2$ suffices, so

$$\delta_{B'}(x_1, x_2) = x_1, x_2 \in B$$

and we take $\phi_<(\bar{x}, \bar{y})$ to be the formula that defines the lexicographic order on pairs.

The constraints of J are encoded in the relations R_g, $g \in \Gamma$. For an m-ary function g, this is defined by a formula ϕ_{R_g} in the free variables $(x_1, \mu_1, \ldots, x_m, \mu_m, e, \nu)$ where each (x_i, μ_i) ranges over elements of U, and (e, ν) ranges over elements of E. To be precise, we define the formula by:

$$\begin{aligned}\phi_{R_g} = \bigvee_{f \in \Gamma'} \Big(& \exists \bar{y} \in V^{ar(f)} : R_f(\bar{y}, e) \wedge \nu \in \mathcal{C}_f \\ & \wedge \bigvee_{e' = (\rho, g, r) \in C_f} \left(\nu = \ con(e') \wedge \bigwedge_{i : \rho_i \in \bar{v}_f} (x_i = e \wedge \mu_i = var(\rho_i)) \right. \\ & \left. \wedge \bigwedge_{i : \rho_i \notin \bar{v}_f} (x_i = y_i \wedge \mu_i = 0) \right) \Big) .\end{aligned}$$

Finally, we define the weight relations. The weight of a constraint $\bar{e} = (e_1, e_2)$ is assigned the product of the weight of $e_1 \in C$ and the weight of $e_2 \in \hat{C}$. We have

$$\phi_{W_N}(\bar{e}, \bar{\beta}) = \bigvee_{e' \in \hat{C}} e_2 = con(e') \wedge \mathrm{MULT}_w(W_N(e_1, \cdot), \bar{b}eta),$$

where w is the numerator of the weight of the constraint e'. The definition of the denominator relation is analogous. □

The next lemma similarly establishes that the reduction in Lemma 4 can be realised as an FPC interpretation.

Lemma 13. *Let Γ and Γ' be valued languages over domain D of finite sizes such that $\Gamma' \subseteq \Gamma_{\equiv}$. Then* $\mathrm{VCSP}(\Gamma') \leq_{\mathrm{FPC}} \mathrm{VCSP}(\Gamma)$.

Proof. Note that adding constants to the value of constraints does not change the optimal solution of the instance. Hence, we only need to adapt to the scaling of the constraint functions. This can be achieved by changing the weights accordingly.

Let $I = (V, C)$ be an instance of $\mathrm{VCSP}(\Gamma')$, given as the relational structure $\mathbf{I} = (\mathrm{dom}(\mathbf{I}), (R_f)_{f \in \Gamma'}, W_N, W_D)$. We aim to construct an instance $J = (U, E)$ of $\mathrm{VCSP}(\Gamma)$ with the same optimal solution.

The set of variables of J is V. For any $f \in \Gamma'$ we fix a function $S(f) \in \Gamma$ such that $S(f) \equiv f$. Then, the formula $\phi_{R_g}(\sigma, d) = \bigvee_{f \in \Gamma'; g = S(f)} R_f(\sigma, d)$ defines the constraints of J. Let $d = (\sigma, g, r)$ be any constraint in E, and $c = (\sigma, f, q)$ be the corresponding constraint in C where $g = S(f)$, and $g = a \cdot f + b$ for some $a, b \in \mathbb{Q}$. We then set the weight r of the constraint d to be $a \cdot q$. This can again be defined by a formula in FPC. □

Next, we show that there is a definable reduction from VCSP(Γ) to the problem defined by a core of Γ.

Lemma 14. *Let Γ be a valued language over D, and Γ' a core of Γ. Then,* VCSP(Γ) $\leq_{\text{FO}}$ VCSP(Γ').

Proof. Since the functions in Γ' are exactly those in Γ, only restricted to some subset of D, we can interpret any instance of VCSP(Γ) directly as an instance of VCSP(Γ'). Since the optimum of both instances are the same, by Lemma 7, this constitutes a reduction. □

The next two Lemmas together show that VCSP(Γ) and VCSP(Γ_c) are FPC-equivalent. The proof follows closely the proof from [9] that they are polynomial-time equivalent.

Lemma 15 (Lemma 2, [9])**.** *Let Γ be a core over domain D. There exists an instance I_p of* VCSP(Γ) *with variables $V = \{x_a \mid a \in D\}$ such that $h_{id}(x_a) = a$ is an optimal solution of I_p and for every optimal solution h, the following hold:*

1. *h is injective; and*
2. *for every instance I' of* VCSP(Γ) *and every optimal solution h' of I', the mapping $s_h \circ h'$ is also an optimal solution, where $s_h(a) := h(x_a)$.*

Lemma 16. *Let Γ be a core over a domain D of finite size. Then,* VCSP(Γ_c) $\leq_{\text{FPC}}$ VCSP(Γ).

Proof. Let $I_c = (V_c, C_c)$ be an instance of VCSP(Γ_c), and let $I_p = (V_p, C_p)$ be an instance of VCSP(Γ) that satisfies the conditions of Lemma 15. We construct an instance $I = (V, C)$ of VCSP(Γ) as follows. The set of variables V is

$$V := V_c \dot\cup V_p = V_c \dot\cup \{x_a \mid a \in D\}.$$

By Definition 8, each function $f \in \Gamma_c$ is associated with some function $g = \gamma(f) \in \Gamma$, such that f is obtained from g by fixing the values of some set of variables of g. Let T_f be the corresponding index set, $t_f : T_f \to D$ the corresponding partial assignment of variables of g, and s_f the injective mapping between parameter positions of f and g. Then, we add for each constraint $c' = (\sigma', f, q) \in C_c$ the constraint $c = (\sigma, g, q)$ to C, where we replace each parameter of g that is fixed to $a \in D$ by the variable x_a, or formally, $\sigma_i = x_{t_f(i)}$ if $i \in T_f$, and $\sigma_i = \sigma'_{s_f^{-1}(i)}$ otherwise. Additionally, we add each constraint of C_p to C with its weight multiplied by some sufficiently large factor M such that every optimal

solution to I, when restricted to $\{x_a \mid a \in D\}$, constitutes also an optimal solution to I_p. For instance, M can be chosen as $M := \sum_{(\sigma,g,q)\in C\setminus C_p} q \cdot \max_{f\in\Gamma_c;\bar{x}} f(\bar{x})$. Note that since the domain and the constraint language are finite, and the functions are finite valued, the value of $\max_{f\in\Gamma_c;\bar{x}} f(\bar{x})$ exists and is a constant. Together, the set of constraints C is defined as

$$C = \{(\sigma, g, q) \mid \exists \sigma', f : g = \gamma(f), (\sigma', f, q) \in C_c,\ \forall i \in T_f : \sigma_i = t_f(i),\ \forall i \notin T_f : \sigma_i = \sigma'_{s_f^{-1}(i)}\}$$
$$\cup \{(\sigma, g, M \cdot q) \mid (\sigma, g, q) \in C_p\}.$$

In order to see that this construction is a reduction, consider the optimal solutions of I_c. Each such optimal solution h_c gives rise to an optimal solution h of I, where $h(x) = h_c(x)$ for $x \in V_c$, and $h(x) = h_{id}(x)$ for $x \in V_p$. In the other direction, let h be an optimal solution to I, and its restriction to V_p, $h_p := h_{|V_p}$ is an optimal solution to I_p. By Lemma 15, the operation s_{h_p} is a permutation on D, and in particular, by repeatedly applying the second part of Lemma 15, the inverse permutation $s_{h_p}^{-1}$ is an optimal solution to I_p as well. Now, again by application of the second part of Lemma 15, we can obtain an optimal solution $h' := s_{h_p}^{-1} \circ h$ to I, for which $h'(x_a) = a$ for each $a \in D$. That means, the restriction of h' to V_c is an optimal solution to I_c.

We now formulate the above construction as an FPC interpretation.

Let I_c be given as a structure $\mathbf{I}_c$ over $\tau_{\Gamma_c} = (<, (R_f)_{\Gamma_c}, W_N, W_D)$. Furthermore, let $I_p = (V_p, C_p)$ be some fixed instance of VCSP(Γ) that satisfies the conditions of Lemma 15. We construct an FPC-interpretation $\Theta = (\bar{\delta}, \varepsilon, \phi_<, (\phi_{R_f})_{f\in\Gamma}, \phi_{W_N}, \phi_{W_D})$ that defines $\mathbf{I} = \Theta(\mathbf{I}_c)$. The universe $\mathrm{dom}(\mathbf{I}_c)$ is the three-sorted set $V_c \dot\cup C_c \dot\cup B_c$. In the same way, the universe of the structure $\mathbf{I}$ is a three sorted set $V \dot\cup C \dot\cup B$. Just as in the proof of Lemma 12, to code elements of V_p and C_p, we fix bijections $var : V_p \to \{1, \ldots, |V_p|\}$ and $con : C_p \to \{1, \ldots, |C_p|\}$

The set V is then defined by the formula

$$\delta_V(x) = x \in V_c \vee x \in \{1, \ldots, |V_p|\}.$$

Similarly, we define C by

$$\delta_C(x) = x \in C_c \vee x \in \{1, \ldots, |C_p|\}.$$

The set of bit positions is chosen to be large enough to encode all weights. We can choose $B = B_c^2$.

$$\delta_B(x_1, x_2) = x_1, x_2 \in B_c,$$

and let $\phi_<$ define the lexicographic order on B_C^2.

For each m-ary function $g \in \Gamma$, we have the formula

$$\phi_{R_g}(\bar{x}, c) = \bigvee_{e=(\rho,g,r)\in C_p} \left(c = con(e) \wedge \bigwedge_{1\le i\le m} x_i = var(\rho_i) \right)$$
$$\bigvee_{f:\gamma(f)=g} \left(\exists \bar{y} \in V_c^{\mathrm{ar}(f)} : R_f(\bar{y}, c) \wedge \bigwedge_{i\in T_f} x_i = var(t_f(i)) \bigwedge_{i\notin T_f} x_i = y_{s_f^{-1}(i)} \right).$$

The weights are given by

$$\phi_{W_N}(c,\bar{\beta}) = (c \in C_c \wedge W_N(c,\beta)) \vee \bigvee_{e=(\rho,g,r)\in C_p} (c = con(e) \wedge \mathrm{MULT}_{r\cdot L}(B_c,\bar{\beta})),$$

where L is given by

$$L = \max_{f\in\Gamma_c;\bar{x}\in D^{ar(f)}} f(\bar{x}).$$

The denominator is given by

$$\phi_{W_D}(c,\bar{\beta}) = (c \in C_c \wedge W_D(c,\beta)) \vee \bigvee_{e\in C_p} (c = con(e) \wedge \mathrm{BIT}(1,\beta)).$$

Here, another case distinction is in place. Either we have $c \in C_c$, and the weight is simply the same as given by W_N and W_D. Or, the constraint c corresponds to some constraint $e = (\rho, g, r) \in C_p$, and we assign the weight $L \cdot 2^{|B_c|} \cdot r$ to c. □

4 Expressibility Result

The fact that VCSP(Γ) is definable in FPC whenever Γ_c does not have the (XOR) property is obtained quite directly from Theorems 10 and 11. Here we state the result in somewhat more general form.

Theorem 17. *For any valued constraint language* Γ *over a finite domain* D*, there is an* FPC *interpretation* Θ *of* $\tau_{\mathbb{Q}}$ *in* τ_Γ *that takes an instance* I *to a representation of the optimal value of* BLP(I)*.*

Proof. We show that it is possible to interpret BLP(I) as a τ_{LP}-structure in I by means of an FPC-interpretation. The statement then follows by Theorem 11 and the composition of FPC-reductions.

Let $I = (V, C)$ be given as the τ_Γ structure $\mathbf{I}$ with universe $\mathrm{dom}(\mathbf{I}) = V \dot\cup C \dot\cup B$. Our goal is to define a τ_{LP}-structure $\mathbf{P}$ representing BLP(I) in given by $(A, \bar{b}, \bar{c})$. The set of variables of $\mathbf{P}$ is the union of the two sets

$$\lambda = \{\lambda_{c,\nu} \mid c = (\sigma, f, q) \in C, \nu \in D^{|\sigma|}\}$$

and

$$\mu = \{\mu_{x,a} \mid x \in V, a \in D\}.$$

In order to refer to elements of D in our interpretation, we fix a bijection $dom : D \to \{1, \ldots, |D|\}$ between D and an initial segment of the natural numbers.

Then, the sets λ and μ are defined by

$$\lambda(c,\bar{\nu}) = \bigvee_{f\in\Gamma} \left(\exists \bar{y} \in V^{\mathrm{ar}(f)} : R_f(\bar{y}, c) \wedge \bigwedge_{1\le i\le ar(f)} \bigvee_{a\in D} \nu_i = dom(a) \right).$$

Here, we assume that $\bar{\nu}$ is a tuple of number variables of length $\max_{f\in\Gamma} \mathrm{ar}(f)$. This creates some redundant variables, related to constraints whose arity is less than the maximum. We also have

$$\mu(x,\alpha) = x \in V \wedge \bigvee_{a\in D} y = dom(a).$$

For the set of linear constraints, we observe that the constraints resulting from the equalities of the form (2) can be indexed by the set

$$J_{(2)} = \{j_{c,i,a,b} \mid c = (\sigma, f, q) \in C, i \in \{1,\dots,|\sigma|\}, a \in D, b \in \{0,1\}\},$$

since we have for each $c \in C$, $i \in \{1,\dots,|\sigma|\}$, and $a \in D$ a single equality, and hence two inequalities, one for each value of b. This can be expressed by

$$\begin{aligned} J_{(2)}(c,\iota,\alpha,\beta) =& c \in C \wedge \bigvee_{f\in\Gamma} \exists \bar{y} \in V^{ar(f)} : R_f(\bar{y}, c) \\ & \wedge \iota \leq ar(f) \\ & \wedge \bigvee_{a\in D} \alpha = dom(a) \\ & \wedge \beta \in \{0,1\}. \end{aligned}$$

Similarly, the constraints resulting from (3) can be indexed by

$$J_{(3)} = \{j_{x,b} \mid x \in V, b \in \{0,1\}\}.$$

Or, as a formula,

$$J_{(3)}(x,\beta) = x \in V \wedge \beta \in \{0,1\}.$$

Finally, we have two inequalities bounding the range of each variable, indexed by

$$J_{(4)} = \{j_{v,b} \mid v \in \lambda \cup \mu, b \in \{0,1\}\},$$

defined by

$$J_{(4)}(\bar{v},\beta) = \lambda(\bar{v}) \vee \mu(\bar{v}) \wedge \beta \in \{0,1\}.$$

The universe $\mathrm{dom}(\mathbf{L})$ is then the three-sorted set $Q \dot\cup R \dot\cup B'$ with index sets Q and R for columns and rows respectively, and a domain for bit positions B', defined by

$$\begin{aligned} \delta_Q(\bar{x}) &= \lambda(\bar{x}) \vee \mu(\bar{x}), \\ \delta_R(\bar{x}) &= J_{(2)}(\bar{x}) \vee J_{(3)}(\bar{x}) \vee J_{(4)}(\bar{x}), \\ \delta_{B'}(x) &= x \in B. \end{aligned}$$

The entries in the matrix $A \in \mathbb{Q}^{Q\times R}$, and the two vectors $\bar{b} \in \mathbb{Q}^Q$ and $\bar{c} \in \mathbb{Q}^R$ consist only of elements of $\{0,1,-1\}$ and the weight of some constraint in C. It is easily seen that these can be suitably defined in FPC. □

Combining this with Theorem 10 yields immediately the positive half of the definability dichotomy.

Corollary 18. *If Γ is a valued constraint language such that property* (XOR) *does not hold for Γ_c, then* VCSP(Γ) *is definable in* FPC.

5 Inexpressibility Result

We now turn to the other direction and show that if VCSP(Γ) is such that Γ_c has the (XOR) property then VCSP(Γ) is not definable in FPC. In fact, we will prove the stronger inexpressibility result that those VCSPs are not even definable in the stronger logic C^ω.

Our proof proceeds as follows. The main result in [12] characterizes the intractable constraint languages Γ as exactly those languages whose extension Γ_c has the property (XOR), by constructing a polynomial time reduction from MAXCUT to VCSP(Γ). We show that this reduction can also be carried out within FPC. It is then left to show that MAXCUT itself is not definable in C^ω. To this end, we describe a series of FPC-reductions from 3-SAT to MAXCUT which roughly follow their classical polynomial time counterparts. Finally, results of [4] and [2] establish that 3-SAT is not definable in C^ω, concluding the proof.

We consider the problem MAXCUT, where one is given an undirected graph $G = (V, E)$ along with a weight function $w : E \to \mathbb{Q}^+$ and is looking for a bipartition of vertices $p : V \to \{0, 1\}$ that maximises the payout function $b(p) = \sum_{(u,v)\in E; p(u)\neq p(v)} w(u, v)$. In the decision version of the problem, an additional constant $t \in \mathbb{Q}^+$ is given and the question is then whether there is a partition p with $b(p) \geq t$.

An instance of (decision) MAXCUT is given as a relational structure $\mathbf{I}$ over the vocabulary $\tau_{\text{MAXCUT}} = (E, <, W_N, W_D, T_N, T_D)$. The universe $\text{dom}(\mathbf{I})$ is a two-sorted set $U = V \dot\cup B$, consisting of vertices V, and a set B of bit positions, linearly ordered by $<$. In addition to the edge relation $E \subseteq V \times V$, there are two weight relations $W_N, W_D \subseteq V \times V \times B$ which encode the numerator and denominator of the weight between two vertices. Finally, the unary relations $T_N, T_D \subseteq B$ encode the numerator and denominator of the threshold constant of the instance.

Lemma 19. *Let* Γ *be a language over* D *for which* (XOR) *holds. Then,* MAXCUT $\leq_{\text{FPC}}$ VCSP($\langle\Gamma\rangle_\equiv$).

Proof. Let $I = (V, E, w, t)$ be a given MAXCUT instance. We define an equivalent instance $J = (U, C, t')$ of VCSP($\Gamma_\equiv$) as follows. Since (XOR) holds for Γ, there are two distinct elements $a, b \in D$ for which $\langle\Gamma\rangle_\equiv$ contains a binary function f, such that $f(a, b) = 1$ if $a = b$ and $f(a, b) = 0$ otherwise. By creating a variable for each vertex in V and adding a constraint $((u, v), f, w(e))$ for each edge $e = (u, v) \in E$, we obtain a VCSP with the same optimal solution. The threshold constant t' is then set to $t' = M - t$, where $M := \sum_{e\in E} w(e)$.

We now define a FPC-interpretation Θ of $\tau_{\langle\Gamma\rangle_\equiv}$ in τ_{MAXCUT} that carries out the construction. Let $\mathbf{I}$ be the relational representation of I over τ_{MAXCUT} with the two-sorted universe $V \dot\cup B$.

The structure $\mathbf{J} = \Theta(\mathbf{I})$ has a three-sorted universe $\text{dom}(\mathbf{J}) = U \dot\cup C \dot\cup B'$ consisting of variables $U = V$, constraints $C = V^2$, and bit positions $B' = B \times \{1, \ldots, |E|\}$.

$$\delta_U(x) = x \in V,$$

$$\delta_C(x_1, x_2) = x_1, x_2 \in V,$$

$$\delta_{B'}(x,\mu) = x \in B \wedge \mu \leq \#_{y,z} E(y,z).$$

Since $M \leq |E| \max_{e \in E} w(e)$, and each $w(e)$ can be represented by $|B|$ bits, $|E| \cdot |B|$ bits suffice to represent the threshold $M - t$.

Each edge $e = (u, v)$ gives rise to a constraint $((u, v), e, w(e))$, which is then encoded in R_f.

$$\phi_{R_f}(\bar{x}, \bar{c}) = E(\bar{x}) \wedge \bar{x} = \bar{c}.$$

The weights are simply carried over.

$$\phi_{W_N}(\bar{c}, b) = W_N(\bar{c}, b)$$

$$\phi_{W_D}(\bar{c}, b) = W_D(\bar{c}, b)$$

The threshold is set to $M - t$. As FPC can define any polynomial-time computable function on an ordered domain, it is possible to write formulas ϕ_{T_N} and ϕ_{T_D} defining the numerator and denominator of the threshold $M - t$ on the ordered sort B'.

The remaining relations R_g corresponding to functions in $g \in \langle\Gamma\rangle_{\equiv} \backslash \{f\}$ are simply empty. □

The next ingredient is to show that the classical series of polynomial time reductions from 3-SAT to MAXCUT can also be carried out within FPC. The chain of reductions goes over three steps, the first one reduces 3-SAT to 4-NAESAT (Not All Equal SAT), then 4-NAESAT is reduced to 3-NAESAT, and finally 3-NAESAT is reduced to MAXCUT. We begin with defining the relational representations of these problems.

An instance of 3-SAT is given as a relational structure over the vocabulary $\tau_{3\text{SAT}} = (R_{000}, \ldots, R_{111})$ with eight ternary relations, one for each possible set of negations of literals within a clause (e.g. $(a, b, c) \in R_{000}$ may represent the clause $(a \vee b \vee c)$ while $(a, b, c) \in R_{101}$ may represent $(\neg a \vee b \vee \neg c)$). Similarly, we assume 3-NAESAT instances to be given as structures over $\tau_{3\text{NAESAT}} = (N_{000}, \ldots, N_{111})$, where $(a, b, c) \in N_{000}$ represents the constraint that not all of a, b and c must evaluate to the same value. Finally, a 4-NAESAT instance is represented as a structure over $\tau_{4\text{NAESAT}} = (N_{0000}, \ldots, N_{1111})$, only now with sixteen 4-ary relations encoding the clauses.

Lemma 20. 3-SAT $\leq_{\text{FPC}}$ MAXCUT.

Proof. 3-SAT $\leq_{\text{FO}}$ 4-NAESAT: Let $\mathbf{I} = (V, R_{000}, \ldots, R_{111})$ be any given 3-SAT instance. Consider a 4-NAESAT instance $\mathbf{J} = (U, N_{0000}, \ldots, N_{1111})$ with $V \subset U$, i.e. there is at least one variable in U not contained in V. Furthermore, let $(a, b, c, z) \in N_{ijk0}$ hold if, and only if, $(a, b, c) \in R_{ijk}$ and $z \in U \backslash V$, and let the relations N_{ijk1} be empty. The instance $\mathbf{J}$ is now satisfiable if, and only if, $\mathbf{I}$ is satisfiable: Whenever there is a satisfying assignment for $\mathbf{I}$, the same assignment extended with $z = 0$ for all $z \in U \backslash V$ will also be a satisfying assignment for $\mathbf{J}$. In the other direction, if there is a satisfying assignment for $\mathbf{J}$, there is always a satisfying one that sets $z = 0$ for all $z \in U \backslash V$, since negating every variable does not change the value of a NAE-clause, and each clause only contains one variable in $U \backslash V$. In terms of a FPC-interpretation, this construction looks as follows.

We take as universe $\mathrm{dom}(\mathbf{J})$ the set V^2, and interpret an element (a,a) as representing the variable $a \in V$, and any element $(a,b), a \neq b$ as a fresh variable in $U \backslash V$.

$$\delta_U(x_1, x_2) = x_1, x_2 \in V$$

$$\phi_{N_{ijk0}}(\bar{x}, \bar{y}, \bar{z}, \bar{w}) = R_{ijk}(x_1, y_1, z_1) \wedge w_1 \neq w_2 \wedge \bigwedge_{\bar{v} \in \{\bar{x}, \bar{y}, \bar{z}\}} v_1 = v_2$$

$$\phi_{N_{ijk1}}(\bar{x}, \bar{y}, \bar{z}, \bar{w}) = \text{False}$$

4-NAESAT $\leq_{\text{FPC}}$ 3-NAESAT: Let $\mathbf{I} = (V, N_{0000}, \ldots, N_{1111})$ be an instance of 4-NAESAT. Note that we can split every clause $\text{NAE}(a,b,c,d)$ into two smaller 3-NAESAT clauses $\text{NAE}(a,b,z)$ and $\text{NAE}(\neg z, c, d)$ for some fresh variable z. The following interpretation realises this conversion.

In order to introduce a fresh variable for each clause of the 4-NAESAT instance, the universe of the 3-NAESAT instance will consist of tuples from $V^4 \times \{0,1\}^5$, where the first eight components encode a clause in $\mathbf{I}$ and the last component is a flag indicating whether the element represents a fresh variable or one that appears already in V. The convention is then that an element of the form $(a,a,a,a,0,\ldots,0)$ represents the variable $a \in V$, and an element of the form $(a,b,c,d,i,j,m,n,1)$ represents the fresh variable that is used to split the clause $N_{ijmn}(a,b,c,d)$.

$$\delta(\bar{x}) = \bar{x} \in V^4 \times \{0,1\}^5$$

$$\begin{aligned}\phi_{N_{ij1}}(\bar{x}, \bar{y}, \bar{z}) = \bigvee_{m,n \in \{0,1\}} & \exists u, v \in V : N_{ijmn}(x_1, y_1, u, v) \\ & \wedge \; x_1 = x_2 = x_3 = x_4 \wedge x_5 = \ldots = x_9 = 0 \\ & \wedge \; y_1 = y_2 = y_3 = y_4 \wedge y_5 = \ldots = y_9 = 0 \\ & \wedge \; \bar{z} = (x_1, y_1, u, v, i, j, m, n, 1)\end{aligned}$$

$$\begin{aligned}\phi_{N_{0ij}}(\bar{x}, \bar{y}, \bar{z}) = \bigvee_{m,n \in \{0,1\}} & \exists u, v \in V : N_{mnij}(u, v, y_1, z_1) \\ & \wedge \; y_1 = y_2 = y_3 = y_4 \wedge y_5 = \ldots = y_9 = 0 \\ & \wedge \; z_1 = z_2 = z_3 = z_4 \wedge z_5 = \ldots = z_9 = 0 \\ & \wedge \; \bar{x} = (u, v, y_1, z_1, m, n, i, j, 1)\end{aligned}$$

The remaining relations are defined as empty.

3-NAESAT $\leq_{\text{FPC}}$ MAXCUT: The following construction transforms a given 3-NAESAT instance $\mathbf{I} = (V, N_{000}, \ldots, N_{111})$ into an equivalent (decision) MAXCUT instance $\mathbf{J} = (\mathrm{dom}(\mathbf{J}), E, W_N, W_D, T_N, T_D)$. Let m be the number of clauses in $\mathbf{I}$, and fix $M := 10m$. For each variable $v \in V$, we have two vertices denoted v_0 and v_1, in our graph, along with an edge (v_0, v_1) of weight M. For each tuple $(x,y,z) \in N_{ijk}$ we add a triangle between the vertices x_i, y_j, and z_k with edge-weight 1. Setting the cut threshold to

$t := |V| \cdot M + 2m$ gives us an equivalent instance: If $\mathbf{I}$ is satisfiable, say by an assignment f, then the partition given by $p(v_i) = f(v) + i \bmod 2$ cuts through every edge of the form (v_0, v_1), and through two edges in every triangle, resulting in a payout of $|V| \cdot M + 2m$. On the other hand, any bipartition of payout larger or equal to $|V| \cdot M + 2m$ has to cut through all edges of the form (v_0, v_1), since it can only cut through two edges in each triangle. Hence, any such bipartition induces a satisfying assignment to the 3-NAESAT instance. We use the following FPC-interpretation to realise this construction.

The universe of $\mathbf{J}$ is defined as a two-sorted set $\mathrm{dom}(\mathbf{J}) = U \dot\cup B$, consisting of vertices $U = V \times \{0,1\}$ and bit positions $B = \{1, \ldots, \alpha\}$ for some sufficiently large α. In particular, α has to be chosen larger than $\log_2 t$. Since m is at most $|V|^3$, taking $\alpha = |V|^4$ suffices.

$$\delta_U(x_1, x_2) = x_1 \in V, x_2 \in \{0,1\}$$
$$\delta_B(\bar\mu) = \bigwedge_{1 \le i \le 4} \mu_i \le \#_v v \in V.$$

The edge relation is given by

$$\phi_E(\bar x, \bar y) = x_1 = y_1 \wedge x_2 \neq y_2 \bigvee_{i,j,k \in \{0,1\}} \exists u, v, w \in V : N_{ijk}(u,v,w) \wedge \bar x, \bar y \in \{(u,i),(v,j),(w,k)\}.$$

The edge weights and the cut threshold are defined by

$$\phi_{W_N}(\bar x, \bar y, \beta) = x_1 = y_1 \wedge x_2 \neq y_2 \wedge \mathrm{BIT}(1,\beta) \vee \mathrm{BIT}\left(10 \cdot \sum_{i,j,k \in \{0,1\}} \#_{u,v,w} N_{ijk}(u,v,w), \beta\right),$$

$$\phi_{T_N}(\beta) = \mathrm{BIT}\left((2 + 10 \cdot \#_v v \in V) \cdot \sum_{i,j,k \in \{0,1\}} \#_{u,v,w} N_{ijk}(u,v,w), \beta\right),$$

$$\phi_{W_D}(\bar x, \bar y, \beta) = \mathrm{BIT}(1,\beta),$$
$$\phi_{T_D}(\beta) = \mathrm{BIT}(1,\beta).$$

Note that the weights and the cut threshold are integer, hence the denominator relation are simply coding 1.

□

Lemma 21. 3-SAT *is not expressible in* C^ω.

Proof. Note that a 3-SAT instance $\mathbf{I} = (V, R^{\mathbf{I}}_{000}, \ldots, R^{\mathbf{I}}_{111})$ can also be interpreted as an instance of $\mathrm{CSP}(\Gamma_{3\mathrm{SAT}})$ for $\Gamma_{3SAT} = \{R_{000}, \ldots, R_{111}\}$ and $R_{ijk} = \{0,1\}^3 \backslash (i,j,k)$. Hence, we can apply results from the algebraic classification of CSPs to determine the definability of 3-SAT. In this context, it has been shown in [4] that the algebra of polymorphisms corresponding to Γ_{3SAT} contains only essentially unary operations. It follows from the result in [2] that 3-SAT is not definable in C^ω. □

Theorem 22. *Let* Γ *be a valued constraint language of finite size and let* Γ' *be a core of* Γ*. If* (XOR) *holds for* Γ'_c*, then* VCSP(Γ) *is not expressible in* C^ω*.*

Proof. Assume property (XOR) holds for Γ'_c. By Lemma 19, MAXCUT FPC-reduces to VCSP($\langle\Gamma'_c\rangle_\equiv$). Lemmas 12 to 16 provide a chain of FPC-reductions from VCSP($\langle\Gamma'_c\rangle_\equiv$) to VCSP($\Gamma$). Since C^ω is closed under FPC-reductions, Lemmas 20 and 21 together show that MAXCUT is not definable in C^ω, and hence neither is VCSP(Γ). □

6 Constraint Languages of Infinite Size

In representing the problem VCSP(Γ) as a class of relational structures, we have chosen to fix a finite relational signature τ_Γ for each finite Γ. An alternative, *uniform* representaation would be to fix a single signature which allows for the representation of instances of VCSP(Γ) for arbitrary Γ by coding the functions in Γ explicitly in the instance. In this section, , we give a description of how this can be done. Our goal is to show that our results generalise to this case, and that the definability dichotomy still holds.

Let Γ now be a valued constraint language over some finite domain D. The challenge of fixing a relational signature for instances of VCSP(Γ) is that different instances may use different sets of functions of Γ in their constraints, and hence, we cannot represent each function as a relation in the signature. Instead, we make the functions part of the universe, together with tuples over D of different arities as their input. Let I be an instance of VCSP(Γ) where the constraints use functions from a finite subset $\Gamma_I \subset \Gamma$, and let m be the maximal arity of any function in Γ_I. We then represent I as a structure $\mathbf{I}$ with the multi-sorted universe $\mathrm{dom}(\mathbf{I}) = V \dot\cup C \dot\cup B \dot\cup F \dot\cup T$, where V is a set of variables, C a set of constraints, B a set of numbers on which we have a linear order, F a set of function symbols corresponding to functions in Γ_I, and T is a set of tuples from $D \cup D^2 \cup \ldots \cup D^m$, over the signature $\tau_D = (<, R_{\mathrm{fun}}, R_{\mathrm{scope}}, W_N, W_D, Def_N, Def_D, Enc)$. Here, the relations encode the following information.

- $R_{\mathrm{fun}} \subseteq C \times F$: This relation matches functions and constraints, i.e. $(c, f) \in R_{\mathrm{fun}}$ denotes that $c = (\sigma, f, q)$ is a constraint of the instance for some scope σ and weight q.
- $R_{\mathrm{scope}} \subseteq C \times V \times B$: This relation fixes the scope of a constraint, i.e. $(c, v, \beta) \in R_{\mathrm{scope}}$ denotes that $c = (\sigma, f, q)$ is a constraint for some function f and weight q, where the β-th component of σ is v.
- $W_N, W_D \subseteq C \times B$: This is analogous to the finite case. These two relations together encode the rational weights of the constraints.
- $Def_N, Def_D \subseteq F \times T \times B$: These two relations together fix the definition of some function symbol in F. That is, $(f, t, \beta) \in Def_D$ denotes that the β-th bit of the numerator of the value of f on input t is 1, and similarly for Def_D and the denominator.

- $Enc \subseteq T \times D \times B$: This relation fixes the encoding of tuples as elements in T, i.e. $(t, a, \beta) \in Enc$ denotes that the β-th component of the tuple t is the element $a \in D$.

The above signature allows now for instances I, I' with different sets of functions Γ_I and $\Gamma_{I'}$ to be represented as structures of the same vocabulary. Since the set of function symbols is part of the universe, the relations Def_N, Def_D are required to give concrete meaning to these function symbols.

We now say, for a (potentially infinite) valued constraint language Γ that VCSP(Γ) is *uniformly definable* in FPC if there is an FPC-interpretation of $\tau_{\mathbb{Q}}$ in τ_D which takes an instance $\mathbf{I}$ of VCSP(Γ) to the cost of its optimal solution. Our inexpressibility result, Theorem 22, immediately carries over to this setting as it is easy to construct an FPC reduction from the τ_Γ representation of VCSP(Γ) to the τ_D representation.

Theorem 23. *Let Γ be a valued constraint language and let Γ' be a core of Γ. If* (XOR) *holds for Γ'_c, then* VCSP(Γ) *is not uniformly definable in C^ω.*

For the positive direction, i.e. to show that VCSP(Γ) is *uniformly definable* in FPC in all other cases, we simply need to adapt the proof of Theorem 17 to fit the new representation.

Theorem 24. *Let Γ be a valued constraint language and let Γ' be a core of Γ. If* (XOR) *does not hold for Γ'_c, then* VCSP(Γ) *is uniformly definable in C^ω.*

Proof. We adapt the proof of Theorem 17 for potentially infinite languages Γ. The main challenge is to work around the variable arities of the constraints.

Let Γ be a constraint language over some finite domain D, and let I be an instance of VCSP(Γ), and $\mathbf{I}$ its relational representation in τ_D. Recall that the set of variables of BLP(I) for $I = (V, C)$ is given by the union of the two sets

$$\lambda = \{\lambda_{c,\nu} | c = (\sigma, f) \in C, \nu \in D^{|\sigma|}\}$$

and

$$\mu = \{\mu_{x,a} | x \in V, a \in D\}.$$

These sets can now be FPC defined from $\mathbf{I}$ as follows.

$$\lambda(c, s) = \exists f \in F : R_{\text{fun}}(c, f) \wedge \exists \beta \in B : \text{Ar}_R(c, \beta) = \text{Ar}_{Enc}(s, \beta)$$

and

$$\mu(x, a) = x \in V \wedge a \in T \wedge \text{Ar}_{Enc}(a, 1).$$

Here, we make use of a formula $\text{Ar}(x, \beta)$ by which we mean that the tuple encoded by the element x has the arity β. The formula can be defined as follows.

$$\text{Ar}_R(x, \beta) = \exists v \in V : \left(R_{\text{scope}}(x, v, \beta) \wedge \forall u \in V, \beta' \in B : \beta' \geq \beta \Rightarrow \neg R_{\text{scope}}(x, u, \beta')\right),$$

$$\text{Ar}_{Enc}(x, \beta) = \bigvee_{a \in D} \left(Enc(x, a, \beta) \bigwedge_{a' \in D} \forall \beta' \in B : \beta' \geq \beta \Rightarrow \neg Enc(x, a', \beta') \right).$$

In words, the formulas ensure that x is a tuple element that is used in the structure, that its β-th component is non-empty, and that for any position $\beta' \geq \beta$, the β'-th component of x is not defined in the structure.

The rest of the proof follows closely to the original one in Theorem 17 without substantial changes. □

References

[1] M. Anderson, A. Dawar, and B. Holm. Maximum matching and linear programming in fixed-point logic with counting. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 173–182, 2013.

[2] A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666 – 1683, 2009.

[3] L. Barto and M. Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61, 2014.

[4] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

[5] D. Cohen, M.C. Cooper, P. Jeavons, and A. Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170(11):983 – 1016, 2006.

[6] A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2:8–21, 2015.

[7] H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.

[8] T. Feder and M.Y. Vardi. Computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.

[9] A. Huber, A. Krokhin, and R. Powell. Skew bisubmodularity and valued CSPs. *SIAM Journal on Computing*, 43(3):1064–1084, 2014.

[10] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[11] M. Otto. *Bounded Variable Logics and Counting — A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag, 1997.

[12] J. Thapper and S. Živný. The complexity of finite-valued CSPs. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, pages 695–704. ACM, 2013.